A METHOD OF COMBINING INTERNET PROTOCOLS FOR SESSION SETUP, TEARDOWN AUTHENTICATION, AUTHORIZATION, AND ACCOUNTING USING THE DIFFERENTIATED SERVICES MODEL

Steven R. Donovan

RIC 99 057

A METHOD OF COMBINING INTERNET PROTOCOLS FOR SESSION SETUP, TEARDOWN, AUTHENTICATION, AUTHORIZATION, AND ACCOUNTING USING THE DIFFERENTIATED SERVICES MODEL

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to the field of Internet multimedia communication, and, more particularly, to a method for combining Internet protocols for session setup, teardown, authorization, and accounting in a Internet Protocol (IP) network, which uses the DiffSERV (Differentiated Services) model in order to guarantee Quality of Service (QoS).

2. Description of the Related Art

The invention of the telephone opened an unprecedented era in personal communication. At the present time, the Internet is opening up another era in personal communication, allowing a level of interactivity previously unknown between computers and groups of computers. In the future, these two services will be combined into one seamless communication medium.

However, the concepts underlying the telephone system and the Internet are fundamentally different. The telephone system is *circuit-based*; meaning that, for example, when a call is set up between caller and callee, a dedicated line, or *circuit*, is maintained between

the two, and, when the call is over, the dedicated line is taken down. The Internet is *packet-based*; meaning that, for example, when a user downloads a web page, or receives an e-mail, the data that comprises the web page or e-mail is broken down into *packets* before being transmitted. The individual packets, although they form one web page or one e-mail message, may take entirely different routes between the sender and the destination. The destination computer puts all the packets together to form the web page.

A fundamental problem lies in providing a circuit-based service, such as a telephone call or videoconferencing, over a packet-based network. While the answer may appear simple–digitize and packetize the audio or visual information - the situation is more complicated than it appears. For one thing, an application such as a telephone call requires a constant transmission rate; something the current Internet cannot guarantee. An application such as videoconferencing using MPEG has stringent real-time requirements in order to avoid the displayed motion appearing jerky. These requirements include a variable transmission rate and very little jitter in the packet arrival times. Once again, at present the Internet cannot guarantee these requirements will be met.

One system for addressing these Quality of Service (QoS) issues on the Internet is the DiffServ model, or Differentiated Services architecture (RFC 2475). In DiffServ, packet traffic shaping is implemented by network routers. In order to specify the transmission requirements, DiffServ uses the Type of Service (ToS) bits in the Internet Protocol (IP) packet header (see FIG.

1). Although the ToS field exists in the current protocol IPv4 (Internet Protocol, version 4), most routers do not use or read the bits in the ToS field. DiffServ uses these bits to tell the router the priority of the packet. Because of this, the ToS field in the IP header is referred to as the DS field.

DiffServ is implemented in the following manner: when packet traffic enters a DiffServ network, the packets are classified and possibly conditioned at the network boundary, most likely in an edge router. The DS field will be filled in with the appropriate bits for that type of traffic, which may depend on customer usage, media specification, general policy, etc. The network nodes inside the DiffServ network will read the DS field to determine how to manage incoming packets. For instance, if an edge router recognizes incoming packets as being high priority, the router will classify those packets as high priority in the DS field, and then send those packets inside the network. When those high priority packets reach a network node, the node will forward them before other packets, because the DS field indicates that they are high priority. This example is somewhat of a simplification, for the DS field classification scheme is more complicated than merely high or low priority, and takes into account throughput, delay, jitter, packet loss, and other traffic characteristics. Taken together, these traffic characteristics make up Quality of Service (QoS).

Because DiffServ classifies these packets into different categories, it works only upon "flow aggregates," which refers to a collection of packet flows. In other words, an interior

network node does not know what a packet contains or if that packet is part of a series of packets; the interior node merely treats it as a member of a certain classification of traffic characteristics. This is in contrast to another method of assuring QoS over a network, the Resource ReSerVation Protocol (RSVP). RSVP sets up a path from network node to network node for a particular packet flow. For example, if an end client device wishes to establish a telephone call over the network, the device would use RSVP to establish a path to the callee's end client device through one or more network nodes. The individual network nodes on the path would then know that a particular identified packet flow will require certain traffic conditions, and resources will be reserved for them. When a node receives one of the packets in the series of packets, the node will recognize it and behave accordingly. While DiffServ looks at flow aggregates, RSVP looks at individual "micro-flows."

For the rest of this description, a DiffServ environment will be assumed. This means that the QoS requirements will be handled by edge routers which will tag individual packets appropriately, while interior network nodes will act upon packets based merely on their DS field.

Even assuming the QoS problems are being handled by DiffServ, there are other services automatically handled in a circuit-based environment which are problematic in an IP-based network. A call has to be set up, establishing a connection between the two end devices, and the resources used in an individual call or session must be tracked, for accounting purposes. In addition, there needs to be the capability to have only authorized sessions or calls from

authenticated users. In the Internet framework, these issues are resolved by different protocols that do different things. Although these individual protocols have been developed in detail, there is at present no known method that sets forth how to use them together in a consistent way across the Internet.

Thus, there is a need for linking these protocols together in a consistent and workable way. In particular, there is a need for a method providing an interchange of parameters among protocols between session setup, authorization, policy, and usage reporting that will support IP communications between Internet Service Providers (ISPs), enterprise networks, and individual clients.

SUMMARY OF THE INVENTION

The present invention provides a method for providing an interchange of parameters among protocols for session setup, teardown, authorization, policy, and usage reporting that will support IP communications in a Differentiated Services model environment.

The present invention provides a method for session or call setup, teardown, authorization, policy and usage reporting in a common way of usage, thereby supporting IP communications across the Internet.

The present invention also provides a method to link together the Session Initiation

Protocol (SIP), Common Open Policy Service (COPS), and Open Settlement Policy (OSP) in a Differentiated Services model environment.

These and other objects are achieved by the preferred embodiment of the present invention. In the preferred embodiment, the messages from the Session Initiation Protocol (SIP), Common Open Policy Service (COPS), and Open Settlement Policy (OSP) are interwoven so that session setup, authorization, policy, and usage reporting are all performed concurrently, in one unified sequence of messages. Likewise, the messages from the Session Initiation Protocol (SIP), Common Open Policy Service (COPS), and Open Settlement Policy (OSP) are interwoven so that session teardown, authorization, policy, and usage reporting are all performed concurrently, in one unified sequence of messages.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects, features and advantages of the present invention will become more apparent from the following detailed description when taken in conjunction with the accompanying drawing in which:

FIG. 1 shows an Internet Protocol Header;

FIG. 2 shows the components of a SIP-based network and an overview of initiating a session;

FIG. 3 shows the components of a Common Open Policy Service (COPS) system;

FIG. 4 shows the components of a Open Settlement Protocol (OSP) system;

FIG. 5 shows a session initiation setup according to an embodiment of the present invention; and

FIG. 6 shows a session teardown according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

As stated above, in the prior art there has been no linkage between the individual protocols that provide for call setup, authorization, accounting, and authentication. These steps are taken care of by the following protocols:

Session Initiation Protocol (SIP) - for setting up connections, or calls;

Common Open Policy Service (COPS) - for policy deployment in network elements; and

Open Settlement Protocol (OSP) - for authorization and usage reporting.

These protocols will be discussed in detail below. In these discussions, the terms "client" and "server" will be used in their abstract functional sense, as processes that may be implemented in any sort of device. This means, of course, that some servers and clients may be running in the same device.

a) Session Initiation Protocol (SIP)

SIP is a signaling protocol that allows for initiating and tearing down connections. There are two components in a SIP system: network servers and user agents. A user agent is an end system that acts on behalf of someone who wants to participate in calls. In general, the user agent contains both a protocol client (a user agent client UAC) which initiates a call and a protocol server (user agent server UAS) which responds to a call (see FIG. 2). There are two different type of network servers as well: a proxy server, which receives requests, determines which server to send it to, and then forwards the request; and a redirect server, which receives requests, but instead of forwarding them to the next hop server, tells the client to contact the next hop directly.

The steps in initiating a session are fairly simple: as shown in FIG. 2, (1) the UAC sends an INVITE request to a SIP server, which in this case, is a proxy server. The proxy server will look in its database to determine where to send the INVITE request. Once that is determined, the proxy server sends the INVITE message to the appropriate next hop. In FIG. 2 KP 05/22/02, the next hop is the callee, but, in reality, there could be a number of hops between the proxy server and the callee. If the proxy server is a redirect server, it would inform the UAC what the appropriate next hop is, and let the UAC do the rest. Once (2) the INVITE message finally reaches the callee UAS, (3) the callee UAS responds with an OK message, which (4) is forwarded to the caller UAC. When the caller UAC receives the OK message, indicating the callee has received the INVITE, (5) the UAC sends an ACK message, which, when (6) received, will start the session.

The steps in terminating a session, or teardown, are even more simple: the UAC sends a BYE message, and the UAS sends a message indicating receipt of the BYE message. In SIP, either the UAC or the UAS may send the BYE message terminating a session.

b) Common Open Policy Service (COPS)

COPS is a simple query and response protocol that can be used to exchange information between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs), as shown in FIG. 3. A policy is a combination of rules and services that define the criteria for resource access and usage. In COPS the PEP sends requests, updates, and deletions to the PDP and the PDP returns decisions back to the PEP. The basic message formats for COPS include Requests (REQs), Decisions (DECs), and Report States (RPTs), among many others.

When particular events occur at a PEP, such as the initiation of a session, the PEP will send a REQ to the PDP to determine the policy regarding the session. The REQ may be an Authentication, Authorization, Accounting (AAA) REQ, which is asking that the session be authorized, authenticated, and kept track of for accounting purposes. If the PDP determines the session fits the AAA policy, the PDP will send its decision DEC to the PEP, thus allowing the PEP to allocate the needed resources. The RPT message is used by the PEP to communicate to

the PDP its success or failure in carrying out the PDP's decision, or to report an accounting related change in state.

c) Open Settlement Protocol (OSP)

OSP is used when there is a central clearinghouse for certain policy decisions. As shown in FIG. 4, OSP is the protocol describing communication between the policy server PDP and the clearinghouse server. This is needed in large networks which require multiple policy servers. Among other things, authorization for QoS levels is handled by the clearinghouse server. The clearinghouse server can also be a trust broker between a large number of network providers and the collecting place for usage reports. As an example, if a PEP sends a REQ AAA to a PDP, the PDP sends a message to the clearinghouse server in order to authorize the call or session. This message is in the form of a <AuthReq>, and the clearinghouse server responds with a <AuthRsp>, which may or may not contain an authorization token, which permits the REQ AAA to proceed. In addition, when a connection or session ends, the use of resources for that session or connection must be recorded for accounting purposes. When the policy server PDP de-installs a particular QoS policy, i.e. registers the end of a session, the policy server PDP sends a <UsageInd> message to the clearinghouse server so that the resource usage is recorded as well as monitored. The clearinghouse confirms the <UsageInd> with a <UsageCnf>.

As stated above, these protocols have been extensively defined and implemented, but to

date there has been no common way of usage for combining them. A preferred embodiment of the present invention, as described below, combines these protocols in order to provide a consistent and common manner of usage for IP-based networks using the Differentiated Services model. In the description below of FIG. 5, a session setup according to the preferred embodiment of the invention will be explained in detail. In the description below of FIG. 6, a session teardown will be explained in detail.

Referring to FIG. 5, at the origination end, there is a SIP user agent client UAC which is attempting to start a session, and the UAC has a local SIP proxy server SIP1, a local Policy server POL1, and a local Router R1. At the destination end, there is a SIP user agent server UAS, which the UAC is attempting to call, and the UAS has local SIP proxy server SIP2, a local Policy server POL2, and a local Router R2. Both the UAS and UAC share the same Clearinghouse CH, shown in the middle. Both POL1 and POL2 are acting as PDPs, and SIP1 and SIP2 are their corresponding PEPs. In the preferred embodiment, when the Clearinghouse sends a positive response to a resource usage request, the Clearinghouse also sends an authorization token. The unit receiving the call is the SIP user agent server UAS, which may be running in any type of IP telephone, computer, media device, or gateway. As stated above, both routers R1 and R2 are working based on the DiffServ model. Therefore, the routers will enforce QoS by altering the DS field in incoming session packets.

In general, the call setup request, authorization and policy installation occur as follows:

1) The UAC sends an INVITE message requesting call setup to SIP1;

2) SIP1 sends a REQ AAA message requesting authentication, authorization, and accounting for the UAC SIP session to the local policy server POL1;

3) Local policy server POL1 sends a <AuthReq> message to the clearinghouse server CH;

4) The Clearinghouse server CH responds with a <AuthRsp> authorizing the session and sending an authorization token to POL 1;

5) POL1 sends a DEC message to SIP1, authorizing installation of the session;

6) SIP1 now forwards the INVITE message to SIP2;

7) SIP2 sends a REQ AAA message requesting authentication, authorization, and accounting for the SIP session to the local policy server POL2;

8) Local policy server POL2 sends a <AuthReq> message to the clearinghouse server CH;

9) The clearinghouse server CH responds with a <AuthRsp> authorizing the session and sending an authorization token to POL2;

10) POL2 sends a DEC message to SIP2, authorizing installation of the session;

11) SIP2 now forwards the INVITE message to user agent server UAS;

12) UAS responds with a 180 RINGING message, which means the UAS is alerting the user to the session;

13) SIP2 sends a REQ LDP message to POL2. This message requests that the appropriate policy be loaded onto R2 concerning this session; it is a local decision point (LDP) message, because the local policy server POL2 will make this

decision, not the clearinghouse;

14) POL2 sends a DEC message to R2, telling R2 of the appropriate policy for the session packets. Since this is a DiffServ environment, router R2 will enable QoS by filling in the DS field of the session packets appropriately when they arrive at the router R2;

15) R2 responds with a RPT message indicating that the policy was installed;

16) POL2 informs SIP2 with a DEC message to install the same policy;

17) SIP2 now forwards the 180 RINGING message to SIP1;

18) SIP1 sends a REQ LDP message to POL1. This message requests that the appropriate policy be loaded onto R1 concerning this session; it is a local decision point (LDP) message, because the local policy server POL1 will make this decision, not the clearinghouse;

19) POL1 sends a DEC message to R1, telling R1 of the appropriate policy for the session packets. Since this is a DiffServ environment, router R1 will enable QoS by filling in the DS field of the session packets appropriately when they arrive at the router R1;

20) R1 responds with a RPT message indicating that the policy was installed;

21) POL1 informs SIP1 with a DEC message to install the same policy;

22) SIP1 now forwards the 180 RINGING message to UAC;

23) UAS responds with a 200 OK message;

24) SIP2 forwards this message to SIP1;

25) SIP1 forwards this message to UAC;

26) UAC acknowledges with an ACK message;

27) SIP1 forwards the ACK message to the SIP2;

28) SIP2 forwards the ACK message to UAS;

29) The session or connection commences.

The actual sequence of messages is divided between the three protocols: message steps 1, 6, 11, 12, 17, and 22-9 are SIP messages; message steps 2, 5, 7, 10, 13-16, 18-21 are COPS messages; and message steps 3-4 and 8-9 are OSP messages. In this manner, the preferred embodiment of the present invention links the three protocols for call setup, authorization, and accounting. Although the above sequence has been described with a clearinghouse server, the preferred embodiment can work in a system without a clearinghouse. In such a network, the policy server handles most of the clearinghouse tasks, and message steps 3-4 and 8-9 would take place inside the policy server.

FIG. 6 shows the steps of a session teardown according to an embodiment of the present invention. The preferred embodiment also links together the protocols when ending a session, as shown in the following sequence of steps:

1) UAC signals the end of the session with a BYE message;

2) SIP1 forwards the BYE message to SIP2;

3) SIP2 forwards the BYE message to UAS;

4) SIP1 sends a REQ noLDP message canceling the policy given in the original REQ LDP message in message step 18 of the setup message sequence above;

5) POL1 sends a DEC Remove message to R1, telling the router to de-install the policy. Since this is a DiffServ environment, the router, up to this point, has been altering the DS field in each of the session packets that arrived. Now, the router will de-install that policy, and stop looking for this session's packets;

6) R1 confirms the policy de-installation with a RPT message to POL1;

7) POL1 sends a DEC message to SIP1, telling the server to de-install the policy;

8) POL1 sends a <UsageInd> message detailing the resource usage to clearinghouse CH;

9) CH confirms with a <UsageCnf> message;

10) UAS sends a 200 OK message confirming receipt of the BYE message;

11) SIP2 forwards the OK message to SIP1;

12) SIP1 forwards the OK message to UAC;

13) SIP2 sends a REQ noLDP message canceling the policy given in the original REQ LDP message in step 13 of the setup message sequence above;

14) POL2 sends a DEC Rem message to R2, telling the router to de-install the policy. Since this is a DiffServ environment, the router, up to this point, has been altering the DS field in each of the session packets that arrived. Now, the router will de-install that policy, and stop looking for this session's packets;

15) R2 confirms the policy de-installation with a RPT message to POL2;

16) POL2 sends a DEC message to SIP2, telling the server to de-install the policy;

17) POL2 sends a <UsageInd> message detailing the resource usage to CH; and

18) CH confirms with a <UsageCnf> message;

As with the setup message sequence described above, the actual sequence of messages is divided between the three protocols: message steps 1, 6, 11, 12, 17, and 22-9 are SIP messages; message steps 2, 5, 7, 10, 13-16, 18-21 are COPS messages; and message steps 3-4 and 8-9 are OSP messages. In this manner, the preferred embodiment of the present invention links the three protocols for call tear-down and usage reporting. Although this has been described with a clearinghouse server, the preferred embodiment can work in a system without a clearinghouse. In such a network, the policy server handles most of the clearinghouse tasks, and message steps 3-4 and 8-9 would take place inside the policy server.

While an embodiment of the present invention has been shown and described, it is to be understood that many changes and modifications may be made thereunto without departing from the spirit and scope of the invention as defined in the appended claims.